Score:_____ Name:_____

ECE 3055 Test 1

1. (20 *points*) A supercomputer has two adders that can be used for addition operations. One adder is not pipelined and it takes 4ns. to add two numbers. The other adder is pipelined and it has a five-stage pipeline with a 2.1ns. clock. The supercomputer has hardware to repeat the add operation automatically for arrays of integers. It can do the entire operation:

```
For I=1 to N
  C[I] = A[I] + B[I]
End;
```

in hardware using one instruction. This is called a vector add instruction. The array values are held in a special high-speed register bank. These registers are fast enough to supply new values to pipeline every clock cycle. The compiler uses the fastest adder depending on the value of N, where N is the number of pairs of array values to add. At which value of N should the compiler switch adders?

5 pts

For N > 4 the compiler should use the pipelined adder, since it will be faster.

N	Total Add Time on non-pipelined adder	Total Add Time on pipelined adder
1	4 ns. (5 pts)	10.5 ns. (10 pts)
2	8 ns.	12.6 ns.
3	12 ns.	14.7 ns.
4	16 ns.	16.8 ns.
5	20 ns.	18.9 ns.
6	24 ns.	21 ns.
7	28 ns.	23.1 ns.
8	32 ns.	25.2 ns.
9	36 ns.	27.3 ns.

2. (*30 points*) The following sequence of MIPS instructions is clocked into the pipeline shown on page 472-476. Examine this figure carefully to see exactly where each signal is located (i.e. before or after pipeline registers). After Clock cycle 5, Indicate the resulting register values in the spaces provided below. All numbers are in hex. Memory location 1 contains AAAAAAAA. Assume that each register contains a value equal to the register number prior to execution of this code.

```
ori     $2,$2,$4
lw      $2,1
add     $3,$1,$4
and     $6,$7,$8
sll     $0,$0,0
```

00000 00 00 111 01 000 00 011 0 10 100

0 0 E 8 3 0 2 4

Instruction = 00E83024

Read Data 1 = 00000007

Read Data 2 = 00000008

3pts each

ALU Result = 0000005

(Data Memory) Read Data = AAAAAAAA

Write Register (Address) = 2

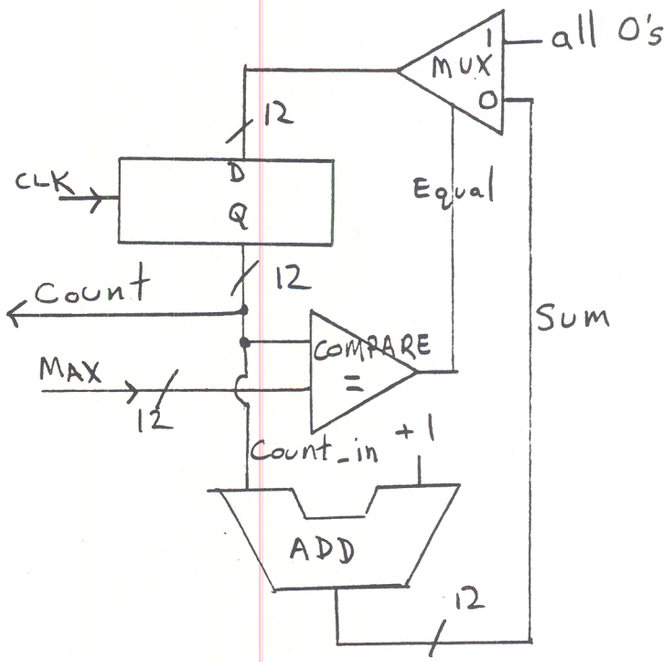
Write Data (input at register file after mux) = 00000006

Memread = 1

RegWrite = 1

Memtoreg = 0

3. *(20 points)* Write a complete VHDL synthesis model for the digital hardware shown in the block diagram.



pts
lib 2
Ent 2
FF 5
Mux 4
Comp 5
Signals 2

Inputs: CLK, MAX
Output: COUNT
Internal Signals:
Equal, Count_in, Sum

```
library IEEE;
use  IEEE.STD_LOGIC_1164.all;
use  IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity counter is
   port( signal Max : in std_logic_vector(11 downto 0);
         signal Count : out std_logic_vector(11 downto 0);
         signal clk: in std_logic);
end counter;
architecture behavior of counter is
   signal  Equal : std_logic;
   signal  Sum, Count_in: std_logic_vector(11 Downto 0);
begin
 equal <= '1' when Max=Count_in else '0';
 count <= count_in;
 process
    begin
    wait until clk'EVENT and clk='1';
    if equal='1' then
        count_in <= "000000000000";
    else
        count_in <= count_in + 1;
    end if;
 end process;
end behavior;
```

4. (*30 points*) The program below is executed on the 5 stage pipelined MIPS described in chapter 6. Answer the following questions about this program.

```
loop:  sw    $2,100($0)
       lw    $3,200($0)
       or    $7,$3,$7
       add   $8,$7,$5
       sub   $5,$3,$2
       sw    $5,100($7)
       beq   $6,$0,loop
```

Part I (*10 points*) If the control unit does not have any hazard detection, forwarding or branch flushing, rewrite the code sequence by adding the minimum number of NOP instructions to eliminate all potential data and branch hazards. Assume other non-NOP instructions follow the branch in the original code sequence above.

```
sw $2,100($0)
lw $3,200($0)
nop
nop
nop
or $7,$3,$7
nop
nop
nop
add $8,$7,$5
sub $5,$3,$2
nop
nop
nop
sw $5,100($7)
beq $6,$0,loop
nop
```

Part II (*10 points*) If the control unit has hazard detection with automatic stalls and automatic branch flushing without any forwarding unit, determine the number of clock cycles required to complete the first loop execution (i.e. start at loop, branch back to loop, and start the lw instruction on the next clock) of the original code sequence. Include the time required to fill and flush the pipeline.

If there were no hazards or branch flushing the program would require 11 clock cycles for execution.

But the program stalls and/or flushes the pipeline 10 clock cycles so a total of 21 clock cycles is required for execution.

Part III (*10 points*) If the control unit is improved by adding the forwarding unit as outlined in the text, determine the number of clock cycles required to complete the first loop execution of the original code sequence. Include the time required to fill and flush the pipeline.

If there were no hazards or branch flushing the program would require 11 clock cycles for execution.

But the program stalls and/or flushes the pipeline 2 (lw + br) clock cycles so a total of 13 clock cycles is required for execution.